

Serifos: Workload Consolidation and Load Balancing for SSD Based Cloud Storage Systems

Zhihao Yao
*Computer & Information Technology
Purdue University*

Ioannis Papapanagiotou
*Cloud Database Engineering
Netflix*

Rean Griffith
VMware

Abstract

Achieving high performance in virtualized data centers requires both deploying high throughput storage clusters, i.e. based on Solid State Disks (SSDs), as well as optimally consolidating the workloads across storage nodes. Nowadays, the only practical solution for cloud storage providers to offer guaranteed performance is to grossly over-provision the storage nodes. The current workload scheduling mechanisms used in production do not have the intelligence to optimally allocate block storage volumes based on the performance of SSDs. In this paper, we introduce Serifos, an autonomous performance modeling and load balancing system designed for SSD-based cloud storage. Serifos takes into account the characteristics of the SSD storage units and constructs hardware-dependent workload consolidation models. Thus Serifos is able to predict the latency caused by workload interference and the average latency of concurrent workloads. Furthermore, Serifos leverages an I/O load balancing algorithm to dynamically balance the volumes across the cluster.

Experimental results indicate that Serifos consolidation model is able to maintain the mean prediction error of around 10% for heterogeneous hardware. As a result of Serifos load balancing, we found that the variance and the maximum average latency are reduced by 82% and 52%, respectively. The supported Service Level Objectives (SLOs) on the testbed improve 43% on average latency, 32% on the maximum read and 63% on the maximum write latency.

1 Introduction

Cloud computing is a popular paradigm for the dynamic provisioning of computing and storage resources deployed on virtualized infrastructure. There is a constant need for higher performance (e.g. latency) by the virtualized storage infrastructure. As the cost of SSDs

gradually decreases, cloud storage providers can provide higher throughput, lower access times, and lower power consumption than the legacy storage systems which are based on spinning disks. For example, Amazon Web Services [1] and Rackspace [5] have recently adopted SSDs as the main storage backend to support their elastic block storage service. In some cases, customized SSDs [19] are used to meet the critical requirements of I/O performance.

The problem of scheduling a storage volume in a multi-tenant storage cluster is more complicated than Virtual Machine (VM) scheduling, which focuses primarily on slicing CPU and memory resources across VMs. First, in SSD-based cloud storage systems, the internal characteristics of SSDs, e.g., the activity of garbage collection or the allocation of over-provisioning space [15], cannot be exposed to the virtualization layer. Second, the workload interference in a shared storage medium, especially SSDs, may decrease the performance further. Finally, workload access patterns are not known in advance. Some existing workload schedulers such as the default available capacity scheduler in OpenStack block storage service, Cinder [4], only take into account the available space of the storage hosts. The lack of performance considerations in scheduling decisions can lead to performance degradation and Service Level Objective (SLO) violations.

For compute resources (CPU and Memory), live migration of VMs is used to deal with unexpected workload variations and to alleviate hotspots [8, 25]. However, storage volume migration is more complex and costly [17] as the resources are rather shared and cannot be sliced. For example, workload interference can cause the cluster to underperform, an incorrect migration may affect other tenants, the time to perform a migration is dependent on external factors (e.g. network throughput), and the data may expire or change during migration. Hence in block storage migration the performance of the backend cluster must be predicted to ensure the benefit

prior to executing a migration operation.

With the goal of equalizing the average (and higher percentile) latencies on every SSD based storage host, Serifos dynamically consolidates workloads so that no host is overloaded or underloaded in terms of I/O resources. Serifos extends other I/O load balancing systems like Romano [20] and Basil [9] to include capabilities such as predicting the aggregate latency for any combination of workloads, and dynamically allocating workloads to achieve a global balanced state of I/O. We use linear regression to construct the workload consolidation models based on the write ratio and the block size of the I/O workloads. In Serifos, the consolidation models are able to precisely predict the host-wide latency for heterogeneous hardware. An I/O load balancing algorithm is integrated into Serifos to achieve the global balanced state.

More specifically, Serifos makes the following contributions:

- Serifos is *accurate* enough to predict the average latency for consolidated workloads on SSD based storage hosts. We create workload consolidation modes according to the performance characteristics of SSD device to ensure the prediction accuracy.
- Serifos is *flexible* enough to manage different I/O access patterns on heterogeneous hardware by building machine dependent models.
- Serifos proves to be *robust* at balancing both at the average and the 99th percentile of the read and write latency across the storage infrastructure. Moreover, Serifos demonstrates the ability to support better performance SLOs.

The paper is structured as follows: In Section 2, related work is briefly introduced. Section 3 presents the background on linear regression techniques and the scheduling in OpenStack Cinder, the performance baseline in our work. Section 4 shows the analysis of the performance characteristics of SSD devices and the workload consolidation model we developed. The design of Serifos' modeler and I/O load balancer is presented in Section 5 followed by evaluation results in Section 6. Finally we conclude our work in Section 7.

2 Related Work

Workload management has been an active topic in academia and industry since the emergence of virtualization and cloud technologies. Prior work has focused on a variety of related problems such as performance modeling and prediction, workload consolidation, and SLO optimization.

Starting from the modeling of SSDs, Agrawal et al. [6] studied the hardware layer design tradeoffs that impact the performance of SSDs through a trace driven simulator. The authors pointed out that SSD performance and lifetime is workload-sensitive. Chen et al. [7] and Hu et al. [11] extended the aforementioned work and talked about how the SSD performance is affected by fragmentation, garbage collection, and write amplification. Subsequently, Huang et al. [12] characterized the performance patterns and built black-box models for one workload running on SSDs to predict performance. The authors indicated that the performance of a single workload on SSDs is predictable, but the mean relative error (MRE) on the extended model (8 parameters) could be as high as 20%. On the contrary, by using two workload parameters, Serifos' performance model for one workload decreases the MRE to 5% and 7% on two SSD devices, respectively. Serifos also extends the performance modeling to a shared environment with many concurrent workloads. Finally, we test Serifos with actual enterprise level SSDs.

Some other studies have looked into developing high level scheduling algorithms and workload management systems. Romano [20] is a storage load balancing system designed to optimize the latency performance of traditional spinning disk based virtualized datacenters. It constructs a performance model via linear regression to predict the average latency of workloads, and subsequently performs I/O load balancing. Romano builds on top of systems like Pesto [10] and Basil [9]. Due to the fundamental differences in the internal mechanisms of spinning disks and SSDs, such as the seek time or garbage collection, Romano, Pesto and Basil are not suitable for predicting the performance of workloads on SSDs. Furthermore, one consolidation model for two workloads is built in these legacy systems. This may result in large prediction error when scaling to many concurrent workloads. The goal of Serifos is similar to these systems and extends them to SSDs. Hence the performance pattern is different from spinning disks, which causes the workload parameters used in the modeling steps and the model to be different. Six different consolidation models created in time intervals similar to Romano provide accurate prediction at scale.

Finally, there are management frameworks focusing on providing better SLOs to block storage consumers. Some, e.g., [26, 27], proposed near optimal scheduling algorithms that considered multi-dimensional resources to enable performance-SLO management capabilities for cloud storage backend systems. PriorityMeister [28] focused on providing end-to-end tail latency QoS to meet performance SLOs. By creating an enforcer daemon on the storage and network side, SLOs could be respected. In a similar fashion, Cake [23] enforced SLOs by limit-

ing the outstanding I/Os (OIOs) at multiple tiers of the storage system. Unlike Serifos, none of these frameworks considered the characteristics of the underlying hardware, which limited their effects in heterogeneous environments.

In short, Serifos is the first I/O load balancing system, which is specifically designed for SSD-based shared storage infrastructure. Based on the observation of SSD performance characteristics, six consolidation models per hardware type are able to provide precise latency predictions. The I/O load balancing algorithm in Serifos can deliver significant improvements in system-wide performance while supporting better SLOs. We present our observation and design detail in the next two sections.

3 Background

3.1 Linear Regression

In statistics, regression is an approach for modeling the relationship between a dependent variable P and one or more independent variables denoted X . A generic model that predicts P given X can be expressed as:

$$P = f(X, \beta) + \varepsilon \quad (1)$$

In this work, we mainly use the linear regression shown in Equation 2 to construct the models because of its lower computation complexity and reasonable model accuracy.

$$P = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \varepsilon \quad (2)$$

In order to evaluate the goodness-of-fit for linear regression model, we employ the R-squared metric (R^2) for simple linear models and adjusted R-squared for multiple linear regression. R-squared is a statistical measure of how close the data are to the fitted regression line and it indicates the percentage of the response variable variation that is explained by a linear model. R-squared values fall between 0 and 100%. In general, the higher the R-squared, the better the model fits the data. The adjusted R-squared is a modified version of R-squared that accounts for the number of predictors in the model. The adjusted R-squared value increases only if the new term improves the model more than would be expected by chance.

3.2 OpenStack Cinder

OpenStack [3] is the most heavily used open source cloud computing platform for public and private clouds. It consists of multiple service components providing different Infrastructure as a Service (IaaS) abstractions. The block storage service in OpenStack [4], code name Cinder, provides a service for managing virtualized block

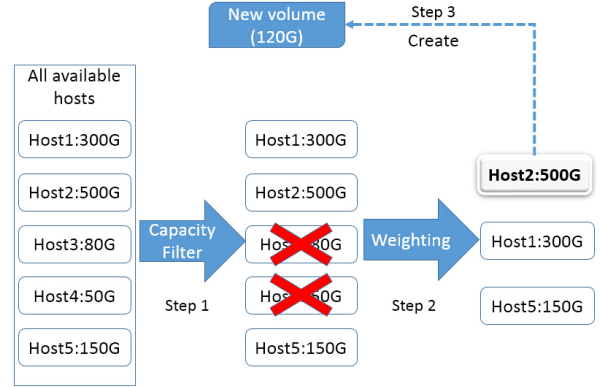


Figure 1: Volume scheduling in OpenStack cinder

storage devices (i.e. volumes) on various storage back-end systems. The cinder-scheduler daemon is responsible for scheduling virtual volumes to physical storage hosts.

The default scheduling algorithm in Cinder is called *Available Capacity* [18]. It is based on a filtering/weighting procedure as shown in Figure 1. The *Available Capacity* scheduler works in three steps. When a new volume request comes in, the scheduler first filters out the hosts which have insufficient resources (e.g. capacity) to meet the needs of the request. Then the scheduler calculates a weight for each of the remaining hosts based on the available capacity, and sorts them based on this weight in a increasing order. The host which has the least weight is chosen as the best candidate to serve the new request. In this work, we use the default *Available Capacity* scheduler of Cinder as the performance baseline, since it is a standard service in OpenStack and many clients will keep the default configuration on their deployment.

4 SSD Performance Modeling

Exploring the workload’s parameters and corresponding latency performance on SSDs provides the essential support for the optimized consolidation algorithm. Previous work by Agrawal et al. [6] and Chen et al. [7] has pointed out that the performance of SSDs is highly workload dependent. Similar to our work, prior studies have focused on the performance model of SSD devices. In our work, we validate some of these findings as well as extend them to a shared environment based on next generation enterprise-level SSDs. Hence, in this section we present our observations on the SSD performance characteristics and consolidation model for concurrent workloads.

Option	Note
direct=1	Bypass OS cache
ioengine=libaio	Linux native asynchronous io library
refill_buffers=1	Refill the I/O buffers on every submit
runtime=60	Every job lasts 1 minute
iodepth=8	I/O concurrency for one workload

Table 1: Fio configuration values

Server name	SSD1	SSD2
Model	Dell R430	
CPU	Intel Xeon E5-2620 v3 2.4GHz	
Memory	8GB	
SSD model	DC S3500	DC S3610
SSD capacity	480GB	400GB
OS	Ubuntu Server 14.04 LTS	

Table 2: Testbed server configuration.

4.1 Methodology

We use three variables as recommended by the Storage Networking Industry Association (SNIA) [14] to represent the access patterns of the workloads on SSDs:

- Write ratio (W): the percentage of write requests in total requests.
- Randomness (X): the percentage of random access I/Os in total requests.
- Block size (S): the number of bytes transferred to/from the storage device. The default unit is kilobyte (KB).

We employed a synthetic workload generator, Fio [2], to generate specific I/O access patterns based on these parameters (W , X and S). We use the Linux native asynchronous I/O library as the default I/O generation engine. To remove the cache effects, the OS cache is bypassed by enabling direct I/O. Some other Fio global configuration values are shown in Table 1. These configurations are referenced from several prior studies [12, 16, 24]. We measure the average latency and use it as the main performance metric – units are microseconds (μs) unless otherwise indicated.

Our testbed consists of two models of SSDs, the Intel DC S3500 series released in 2013, and the Intel DC S3610 series released in 2015. Table 2 describes the detailed configurations of the servers. Each of the servers was equipped with a set of these drives, but we only use one drive to perform the I/O tests. We used the other drive as the OS drive so that the Fio-issued requests are only sent to the test drive. Instead of using the raw device (e.g. `/dev/sdb`), we use LVM (Logical Volume Manager) to create block volumes and connect each volume to the

Fio process. We choose LVM to mimic real-world deployments since it is the default volume driver in OpenStack Cinder.

An important first step in SSD performance measurements is *aging* the device [15]. Typically a fresh SSD exhibits a transient period of elevated performance, which evolves to a stable performance state relative to the workload being applied. It is critical to ensure the SSD under test workload is running at the steady state to collect meaningful results. A method [21], suggested by SNIA, is applied to age the device before running any experiments so that the SSD state at the start of every experiment is in a steady and consistent state. First, the device is purged by issuing the ATA secure erase command. Then we age the device by running a special workload issuing 128KB sequential write I/O requests to the entire logical block addresses for 2X (twice) the user capacity. Each experiment is performed five rounds and the result of the fifth round is used as valid result.

4.2 Device Model

We build a device model to describe the correlation between the significant workload parameters and the average latency of a workload running on the SSDs. We employ linear regression and analysis of variance (ANOVA) to determine whether the three workload parameters are effective in predicting the average latency.

The test vector of each parameter used to describe all possible access patterns of an I/O workload are defined as:

$$W = \{0\%, 10\%, 20\%, 30\%, \dots 90\%, 100\%\} \quad (3)$$

$$X = \{10\%, 20\%, 30\%, \dots 90\%, 100\%\} \quad (4)$$

$$S = \{4, 8, 16, 32, 128, 256\} \quad (5)$$

This test vector consists of 660 different access patterns in total for one workload. Note that we exclude the $X = 0\%$ which represents a 100% sequential workload. This is due to the fact that pure sequential workloads have completely different performance patterns on an SSD compared to random workloads in terms of latency. Similar observations were also made by [6, 7, 12]. In fact, even for a spinning disk based storage systems, an 100% sequential workload would not provide any benefit to performance modeling other than to add a large relative error [9].

We first apply polynomial regression on the measured latency to obtain the most accurate model. The polynomial model contains an intercept, linear terms, interactions and squared terms. Romano [20] found that sometimes the interaction and squared terms show enough impact in the model for spinning disks based storage infrastructure. The polynomial regression results of two

	SSD1		SSD2	
Adj. R^2	0.98		0.991	
factor	Coef	p-value	Coef	p-value
INTCP	-852.76	2.42e-15	-346.32	2.6e-5
W	32.596	2.68e-36	27.815	1.19e-42
X	3.367	1.13e-5	1.8305	0.439
S	29.382	3.88e-125	21.204	3.01e-282
$W : X$	-0.1645	7.68e-16	-0.0424	0.00676
$W : S$	0.2028	1.42e-138	0.0832	6.62e-146
$X : S$	-0.051	2.31e-13	0.0055	0.0529
W^2	-0.246	3.3e-30	0.2761	3.05e-56
X^2	0.045	0.069	0.0001	0.994
S^2	-0.0108	0.0095	-0.0041	1.16e-11

Table 3: The polynomial model for two types of storage server. Terms with high p-values are bolded. INTCP and Coef are the abbreviation for intercept and coefficient respectively.

	SSD1		SSD2	
Adj. R^2	0.94		0.975	
factor	Coef	p-value	Coef	p-value
INTCP	-421.19	2.13e-6	-351.13	1.722e-7
W	13.959	3.1e-39	9.228	1.584e-31
X	-2.934	0.0009	0.4722	0.57
S	33.968	0	23.644	0

Table 4: The linear model for two types of storage server. Terms with high p-values are bolded.

SSD servers are shown in Table 3. Both coefficients and p-values are captured. We derive the p-value from the ANOVA test at the 95% confidence level. It tests the null hypothesis that the coefficient of a term is equal to zero (no effect). If the p-value of a term is greater than 0.05, it suggests that the null hypothesis is accepted and the changes in the predictor are not associated with the changes in the response, and therefore the predictor can be removed from model safely. Terms with high p-value (> 0.05) are shown in **bold** font in the table.

In both models, we notice that the interaction terms and squared terms have very small coefficient. Yet some of them are significant ($p\text{-value} < 0.05$). This indicates that these terms have minimal impact on the average latency on SSD device. Based on this observation, we decide to apply linear regression to build a device model without interaction and squared terms.

Table 4 describes the device model in linear form. The adjusted R-squared values of both models are 0.94 and 0.975 respectively. Therefore, the accuracy of both models does not decrease much due to the exclusion of interaction and squared terms. The linear terms are able to precisely describe the device model. Another obser-

	SSD1		SSD2	
Adj. R^2	0.972		0.986	
factor	Coef	p-value	Coef	p-value
INTCP	583.36	2.285e-14	-275.38	8.929e-11
W_1	-4.105	4.667e-8	4.426	8.3927e-20
W_2	-3.903	3.77e-7	4.495	2.326e-20
S_1	23.26	0	19.663	0
S_2	23.164	0	19.713	0

Table 5: The consolidation models for two workloads

vation from the linear models is that randomness is not a significant factor to the average latency of the SSD2 server model and has a very small coefficient in the SSD1 server model. To illustrate this fact, we analyze the effect on average latency of each workload parameter in detail. When randomness changes from 10% to 100%, the average latency only varies about 0.3ms, given that all other parameters are held constant, whereas the effect of block size is nearly 9ms. Write ratio can affect the average latency by almost 2ms. Hence the randomness may not be a important factor in terms of average latency of SSD device. The linear models without randomness are shown at Equation 6 and Equation 7.

$$L_{avg_SSD1} = -621.06 + 12.96W + 33.97S \quad (Adj.R^2 = 0.94) \quad (6)$$

$$L_{avg_SSD2} = -325.59 + 9.23W + 24.64S \quad (Adj.R^2 = 0.975) \quad (7)$$

The adjusted R-squared values do not change in the simplified models that exclude randomness. Therefore we can conclude that only write ratio and block size have significant contributions to the average latency of the non-sequential workload running on SSD.

4.3 Workload Consolidation Model

In a cloud storage infrastructure, the resources of a single physical host are shared among several concurrent workloads. In order to predict the host-wide average latency, we need to investigate the performance patterns of consolidated workloads.

The methodology is similar to the device model steps. However only the write ratio (W) and block size (S) are used to describe a workload. First, we examine the model for two workloads in which the write ratio and block size are the same as in Equation 3 and 5. The total number of access patterns in the test set is 4356 or $(11 \times 6)^2$. The linear models of two servers are shown in Table 5. Both models fit the data very well according to their adjusted

R-squared values. This proves that the write ratio and block size can effectively be used to predict average latency in the shared environment.

From the consolidation models, we notice that the coefficients of write ratio, W_1 and W_2 , are very close, so are the coefficients of block size, S_1 and S_2 . So there is an opportunity to combine the corresponding parameters of two workloads in the model. We calculate the sum of the write ratio and the sum of the block size, and apply them to build the consolidation model in another form, shown as follows:

$$L_{avg_SSD1} = -583.36 + -4.02(W_1 + W_2) + 23.21(S_1 + S_2) \quad (Adj.R^2 = 0.975) \quad (8)$$

$$L_{avg_SSD2} = -275.38 + 4.46(W_1 + W_2) + 19.68(S_1 + S_2) \quad (Adj.R^2 = 0.986) \quad (9)$$

These models do support the conclusion that there is a strong correlation between the average latency and the workload parameters in a sum format. To verify this observation, we scale the number of concurrent workloads to 3 and 4. We reduce the running time of one test from 1 minute to 30 seconds so that the whole set of tests can be done in couple of days. The linear models of 3 workloads are:

$$L_{avg_SSD1} = 419.21 + -2.51(W_1 + W_2 + W_3) + 24.987(S_1 + S_2 + S_3) \quad (Adj.R^2 = 0.953) \quad (10)$$

$$L_{avg_SSD2} = 34.233 + -0.357(W_1 + W_2 + W_3) + 21.561(S_1 + S_2 + S_3) \quad (Adj.R^2 = 0.972) \quad (11)$$

and 4 workloads:

$$L_{avg_SSD1} = 179.65 + -1.621(W_1 + W_2 + W_3 + W_4) + 24.684(S_1 + S_2 + S_3 + S_4) \quad (Adj.R^2 = 0.941) \quad (12)$$

$$L_{avg_SSD2} = -302.78 + -1.121(W_1 + W_2 + W_3 + W_4) + 21.937(S_1 + S_2 + S_3 + S_4) \quad (Adj.R^2 = 0.983) \quad (13)$$

The coefficients of all workloads are still close enough to be combined into a sum term. The adjusted R-squared values of all models indicate a good fit on measured data. Therefore, the workload consolidation model can be expressed as:

$$L = \beta_0 + \beta_1 \sum_{i=1}^n W_i + \beta_2 \sum_{i=1}^n S_i + \varepsilon \quad (14)$$

where n is the number of concurrent workloads. According to this model, we are able to apply it to predict the host-wide average latency based on the write ratio and block size of every concurrent workload on the shared physical storage host. Moreover, the coefficient of the sum of workloads' block sizes is much larger than the coefficient of the sum of write ratios in all above models. Hence we can conclude that the block size is the major latency predictor in the consolidation model compared with write ratio, although both of them show significance in the model.

5 System Design of Serifos

Based on the workload consolidation model, we design Serifos, an I/O load balancing system for SSD based shared storage infrastructure. The goal of Serifos is to optimize the system-wide latency performance. Serifos is designed to be integrated into OpenStack as a dynamic scheduling plug-in. There are two major components in the Serifos: the *Modeler* and the *Load-balancer*.

Modeler: Before a new server is deployed in a production environment, the modeler runs five synthetic test sets to generate six workload consolidation models to predict the average latency, including five basic models and one extended model. Each test set consists of a combination of a fixed number (1-5) of workloads with different access patterns. One basic workload consolidation model is generated based on collected average latency and related workload combinations from one test set. Five basic models derived from five test sets are responsible for providing the prediction for 1-5 workloads respectively. The extended model is built based on the aggregated measurements of all five test sets, and is applied when there are more than five concurrent workloads. The intercept and coefficients of models will be stored in a JSON file which will be used by the *Load-balancer*. In order to reduce the time to build the model, we used a simplified version of the test vectors, shown in Equation 15 and 16. The evaluation results in Section 6 show that such simplification do not affect the prediction accuracy of all consolidation models.

$$W = \{25\%, 50\%, 75\%\} \quad (15)$$

$$S = \{4, 8, 32, 128\} \quad (16)$$

It is possible that the write ratio is not significant (p-value > 0.05) in the standard form of consolidation model with simplified test vectors. In such cases, the non-significant factor is removed from the model and corresponding coefficient is set to 0. Then the consolidation model is re-built based on the significant factors.

We carefully evaluated the test cases in the test set. We found that some tests of multiple workloads are actually repeated because the order of workloads do not play

a role in terms of performance. For example, the test pattern [w1(25%, 8k), w2(50%, 128k)] and [w1(50%, 128k), w2(25%, 8k)] are the same cases. Such replicated test cases are removed from the test set by generating the mathematical combinations (with replacement) of the test vectors of workloads. Comparing with the calculation of the Cartesian product of the test vectors, the time needed for running all test sets reduces from several weeks to only one and half days (including device preconditioning).

Some previous systems, such as Romano [20] and Basil [9], only create one consolidation model of two aggregated workloads. This model is recursively applied for predicting performance when there is more workloads. In Serifos, with the benefit from test vector simplification and test set de-duplication, we are able to run more complex tests for more workloads within similar time consumption. Five basic models allow very accurate predictions on the basis of real measurements. The idea of the extended model is based on the observations that the block size is the most determined parameter to the average latency and the coefficients of the block size term in Equation 8-13 are very close (~ 24 for SSD1, ~ 21 for SSD2). We make a reasonable assumption that this trend can scale to more concurrent workloads. Therefore, we build an extended model for more than five workloads depending on the aggregated measurements. The evaluation results in Section 6.1 verify our assumption and prove that the extended model is able to precisely predict the average latency.

I/O load Balancer: The I/O load balancer is the major component in the system. It aims to equalize the average latency across multiple storage servers and to reach the global balanced state by re-allocating the existing workloads. Based on the consolidation models built by the *Modeler* module, the average latency can be predicted before performing any workload migration to ensure performance benefits.

Our core balancing algorithm is inspired by the Best Fit Decreasing (BFD) approximation of the well-known NP-hard bin packing problem. In the classical problem, N items with different sizes are placed in a set of bins. The goal is to pack the items in as few bins as possible. Among a various of bin packing algorithms, BFD is able to achieve the best asymptotic worst-case performance [13]. The BFD algorithm is an offline approximate approach with the time complexity of $O(n \log n)$ that sorts items at the beginning according to their size in a decreasing order so that the large items can be processed first. Then each item is placed in the best, i.e. tightest, bin to minimize the empty space. In our scenario, the corresponding attribute of item size is the block size of a workload. Comparing with write ratio, the block size has a much greater impact and is the decisive factor for av-

Algorithm 1 I/O Load Balancing Algorithm

```

procedure LOAD_BALANCE(wkld_list, host_list)
    wkld_list.sortDecreasing(wkld.blockSize)
    foreach wkld in wkld_list do
        best_host = None
        best_perf = MAX_PERF
        foreach host in host_list do
            if host.freeCapacity < wkld.size then
                continue
            end if
            perf = PredictPerf(host, wkld)
            if perf < best_perf then
                best_host = host
                best_perf = perf
            end if
        end for
        if best_host  $\neq$  currentHost then
            scheduleMigration(currentHost,
                best_host)
        end if
    end for
end procedure

procedure PREDICTPERF(host, wkld)
    model = loadModel(host.numOfWklds + 1)
    new_wr_sum = host.wr_sum + wkld.wr
    new_bs_sum = host.bs_sum + wkld.bs
    perf = model.predict(new_wr_sum, new_bs_sum)
    return perf
end procedure

```

erage latency based on the observation on device model and consolidation model. The best is also defined as the lowest average latency among all predictions.

Algorithm 1 outlines two functions running in the I/O load balancer. The main function *load_balance* takes two arguments as input: the workloads running in the system, and the available storage hosts. First, all workloads are sorted in decreasing order based on their block size, because the block size is the decisive factor to the average latency. Then the algorithm starts at the workload which has the largest block size. For a given workload, the function evaluates the predicted average latency of all available hosts if that workload is allocated on each host, under the assumption that the candidate hosts has enough free capacity to allocate that workload. Then the host having the lowest predicted average latency will be chosen as the best candidate to allocate the workload. If the current host of a workload is not the best host, the workload will be scheduled for a migration operation. The function *PredictPerf* predicts the average latency after a new workload is placed on a host. The write ratio and block size of new workload are added to the corresponding sum of parameter of existing workloads on the

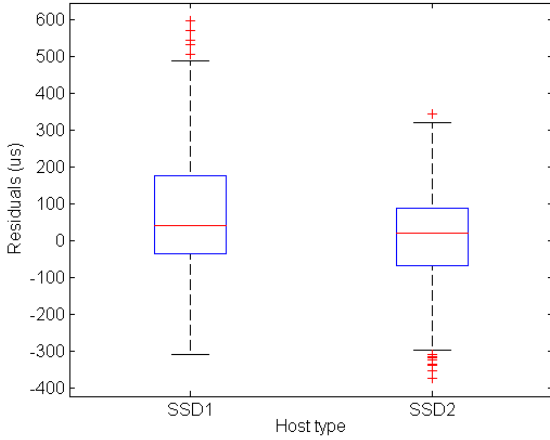


Figure 2: Residuals distribution for 500 tests.

host. After that the consolidation model is able to calculate the predicted average latency.

It is obvious that the key factor impacting the balancing effect is the accuracy of the consolidation model. If the prediction error is not small enough, the balancer may make an incorrect choice of the best candidate host of a workload. As a consequence, the global balanced state can not be achieved, and I/O resources will be wasted on unnecessary workload migrations. This is also the reason we choose to build individual consolidation models for each number of workloads.

6 Evaluation

We implement a prototype of Serifos in Python to build the consolidation model and balance workload performance. We chose Python so that our system can be eventually committed to the OpenStack Cinder project and work with the OpenStack telemetry service (Ceilometer) to retrieve the necessary workload statistics. We also choose to develop a simple volume management framework based on LVM to support the modeler and I/O load balancer component for repeatable and quick evaluation. The default available capacity scheduling algorithm of Cinder, introduced in 3.2 was also implemented in the framework for comparison purposes. In this section, the prediction accuracy of the consolidation models and the I/O load balancing effects are evaluated through several experiments on a real testbed. There are five storage servers in the testbed, 3 of SSD1 type and 2 of SSD2 type (Table 2).

A revised version of test vectors (Equations 17-19) is applied to generate representative workloads [22]. The default randomness for all access patterns is 100%. Size

Model	SSD1	SSD2
1	7%	5%
2	12%	7%
3	12%	7%
4	10%	5%
5	9%	10%
5+	16%	8%

Table 7: Mean relative error of the consolidation models

C is added as a new parameter of workload so that the LVM manager can create the storage volume of workload accordingly. The access pattern of a workload is randomly selected from these test vectors. In order to speed up the testing, workload migration is emulated by moving the Fio process from one server to another instead of actually copying data between different hosts. The attached volume is also deleted on the original server and re-created on the destination server. This helps us reduce the experimental time from weeks to days.

$$W = \{5\%, 30\%, 50\%, 70\%, 95\%\} \quad (17)$$

$$S = \{4, 8, 16, 32, 64, 128, 256\} \quad (18)$$

$$C = \{30, 60, 90, 120\} \quad (19)$$

6.1 Prediction of Consolidation Models

Before evaluating the prediction accuracy, the modeler is executed on one server of each type to construct the consolidation models. The regression results of five basic models (model 1-5) and one extended model (model 5+) for both types of server are shown in Table 6. These models indicate that the block size on SSD1 server is the direct predictor for the average latency. The write ratio only plays a limited role on the consolidation model for 5 and more workloads. On SSD2 server, most models follow the general model form except the model for 2 workloads. However, the write ratio still has minimal impact compared with the block size. These models show that block size is the decisive predictor of the average latency again. Moreover, all of the models' adjusted R-squared values are very high which means the models fit the collected data very closely.

To examine the prediction accuracy of these models, we run 500 tests to evaluate five basic consolidation models (100 each) and 200 tests for extended model on both types of server. Each test on basic models contains a corresponding number of workloads with random access patterns. For the extended model, each test has random number of 6-10 workloads. The mean relative error (MRE) values of all models are listed in Table 7. Six models out of ten basic models have less than 10% MRE and the worst MRE is only 12%. The extended model of

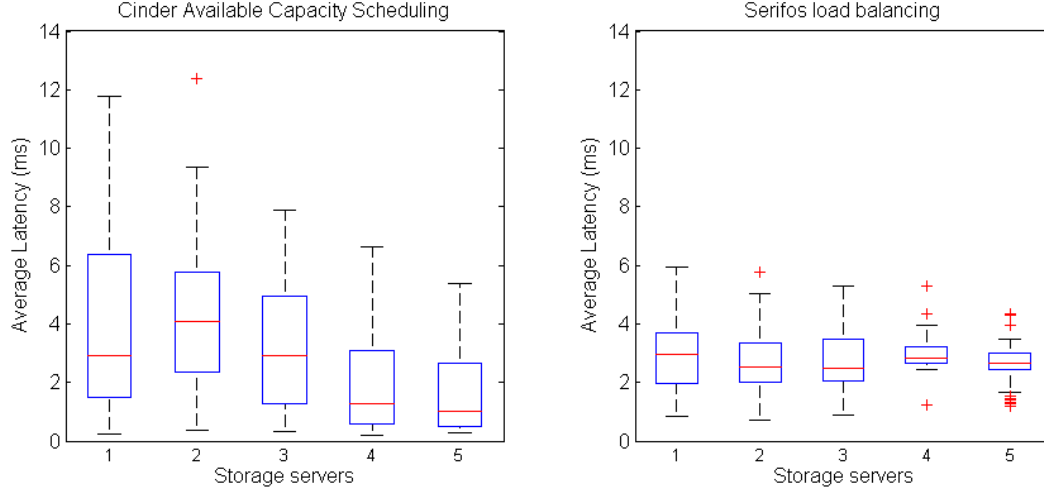


Figure 3: Average latency distribution on 50 test cases.

	SSD1				SSD2			
Workloads	INTCP	$W_{sum.co}$	$S_{sum.co}$	Adj. R^2	INTCP	$W_{sum.co}$	$S_{sum.co}$	Adj. R^2
1	113.44	0	22.135	0.994	216.51	-1.19	19.628	0.999
2	0	0	24.497	0.988	42.669	0	20.691	0.996
3	0	0	24.714	0.981	-86.634	0.533	21.339	0.995
4	81.969	0	23.587	0.977	-188.26	0.907	21.729	0.994
5	0	0.578	23.919	0.98	-133.83	0.519	21.906	0.994
5+	0	0.646	23.913	0.981	-137.81	0.597	21.821	0.995

Table 6: The consolidation models for two servers. If the coefficient is 0, it means the corresponding term is not significant ($p\text{-value} > 0.05$) in the standard linear consolidation model. The model is re-built only based on significant terms and used in load balancing.

SSD1 has 16% MRE while SSD2 server achieves an outstanding accuracy on the newest SSD device with only 8% MRE. To deeply investigate the prediction accuracy, the distribution of basic models' residuals (difference between real and prediction) are presented in Figure 2. The residuals of all basic models are calculated as the prediction error ϵ . The median value of two servers are near 0 and the lower 50% residuals are in the range of [-50, 200] and [-50, 100] approximately. Note that the unit of residuals is microsecond. These results demonstrate that the accuracy of Serifos consolidation models is robust enough to provide precise performance predictions to the I/O load balancing algorithm. Moreover, it validate the fact that our method of building the consolidation models is appropriate for SSD based storage.

6.2 I/O Load Balancing

In this section, we randomly generate 10-16 workloads on the testbed which is managed by the available capacity scheduling algorithm in OpenStack Cinder. Then the

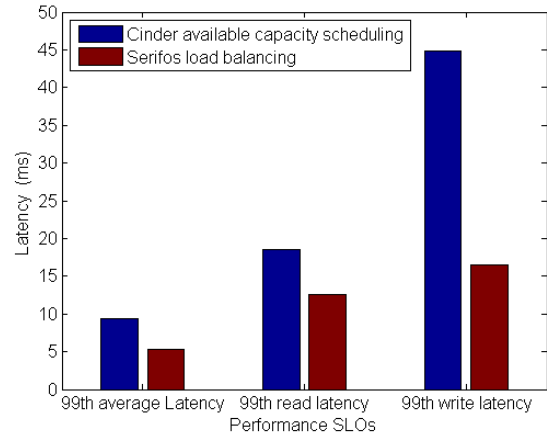


Figure 6: Performance SLO supported by OpenStack Cinder and Serifos.

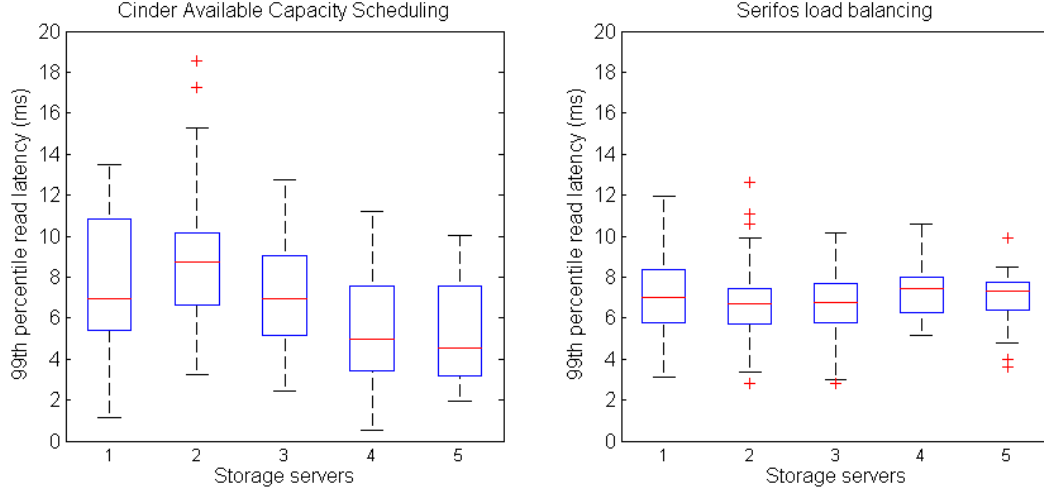


Figure 4: 99th read latency distribution on 50 test cases.

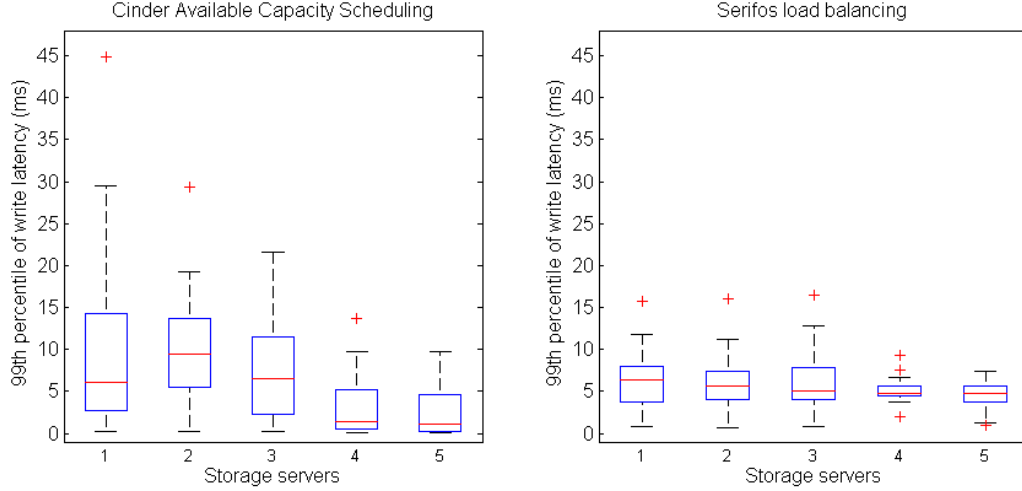


Figure 5: 99th write latency distribution on 50 test cases.

load balancer is enabled to optimize the global latency on our testbed. The experiment is repeated 50 times.

Figure 3 shows the aggregated latency of Cinder scheduling and load balancing. In the results for the baseline Cinder scheduler, shown in the left graph, the first three hosts (type *SSD1*) are overloaded. Because the *SSD* capacity on *SSD1* servers are larger than the *SSD2*, the Cinder scheduler places more storage volumes on *SSD1* servers. Unawareness of the device characteristics leads to an imbalanced state in the left graph. With the performance predictions from accurate consolidation models, the load balancer successfully minimizes the performance difference across the storage hosts, shown in the right graph in Figure 3. The statistic of aggregated

result indicates that the variance of system-wide average latency is reduced by 82%. More importantly, the maximum of measured average latency decreases by 52%, from 12.39ms to 5.95ms, after Serifos load balancing.

In a multi-tenant environment, cloud clients are interested both for the higher percentiles (i.e. 99) of a performance metric and the average. Therefore, besides the average latency, the 99th percentile of read/write latency are also captured and presented in Figure 4 and 5. Not surprisingly, the I/O load balancer reduces the variance of two measurements by 71% and 84% respectively. The median and the lowest 50% of all storage hosts also become much closer in both figures which means there is no overloaded or underloaded host. The majority of

clients are able to have more stable performance after Serifos load balancing.

Finally, we present the supported performance SLO setting by two systems in Figure 6. When the cloud provider offers performance SLOs, they have to ensure that every workload running on the infrastructure should receive no worse than the SLO in normally 99% of service time. Otherwise, a penalty will be applied due to the SLA violation. Better performance SLOs are an important consideration when evaluating the workload management system for cloud. Figure 6 compares the supported SLO value of Cinder and our system for different performance SLO metrics on the same testbed. All three kinds of metric are reduced significantly. After enabling the load balancer, the storage infrastructure is able to support much lower latency SLOs in 99% of service time. More specifically, 43% lower on average latency, 32% lower on maximum read and 63% lower on maximum write latency.

7 Conclusion and Future Work

In this paper, we presented Serifos, an autonomous performance modeling and load balancing system designed for SSD based cloud storage infrastructures.

The first key contribution of Serifos is a workload consolidation model for predicting the average latency for multiple concurrent workloads running on a shared SSD device. After an exhaustive evaluation of workload parameters, our findings indicate that the write ratio and block size of a workload have a strong correlation to the average latency. Moreover, the host-wide average latency can be predicted by the sum of workloads write ratios and the sum of workloads block sizes on a shared host. The six consolidation models built by Serifos are able to precisely predict the average latency of any combination of workloads and are used as guidelines for I/O load balancing.

Second, we integrate a load balancing engine with Serifos. The load balancer has a goal to optimize the system-wide latency performance by dynamically migrating workloads across storage servers. With the help of accurate consolidation models, the migrations recommended by the load balancer deliver a significant improvement in the latency performance of I/O workloads running in the backend storage systems.

We evaluated Serifos on an actual SSD deployment. Our experimental results indicate that Serifos consolidation model is able to maintain the mean prediction error around 10% for heterogeneous hardware. The load balancing engine enhanced the average latency performance of the testbed managed by OpenStack Cinder by 82%. Furthermore, the supported performance SLOs in 99% service time are reduced by 43% on average latency, 32%

on the maximum read, and 63% on the maximum write latency.

In the future, we plan to consider the migration cost in the I/O load balancing algorithm. Some special cases such as migrating a large volume with a small amount of improvement needs to be carefully considered.

References

- [1] Amazon web service. <http://aws.amazon.com/>.
- [2] Fio. <http://github.com/axboe/fio/>.
- [3] Openstack. <https://www.openstack.org/>.
- [4] Openstack cinder. <https://wiki.openstack.org/wiki/Cinder>.
- [5] Rackspace. <http://www.rackspace.com/>.
- [6] AGRAWAL, N., PRABHAKARAN, V., WOBBER, T., DAVIS, J. D., MANASSE, M. S., AND PANIGRAHY, R. Design tradeoffs for ssd performance. In *USENIX Annual Technical Conference* (2008), pp. 57–70.
- [7] CHEN, F., KOUFATY, D. A., AND ZHANG, X. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *ACM SIGMETRICS Performance Evaluation Review* (2009), vol. 37, ACM, pp. 181–192.
- [8] CLARK, C., FRASER, K., HAND, S., HANSEN, J. G., JUL, E., LIMPACH, C., PRATT, I., AND WARFIELD, A. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2* (2005), USENIX Association, pp. 273–286.
- [9] GULATI, A., KUMAR, C., AHMAD, I., AND KUMAR, K. Basil: Automated io load balancing across storage devices. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)* (2010), vol. 10.
- [10] GULATI, A., SHANMUGANATHAN, G., AHMAD, I., WALDSPURGER, C., AND UYSAL, M. Pesto: online storage performance management in virtualized datacenters. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (2011), ACM, p. 19.
- [11] HU, X.-Y., ELEFTHERIOU, E., HAAS, R., ILIADIS, I., AND PLETKA, R. Write amplification analysis in flash-based solid state drives. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference* (2009), ACM, p. 10.
- [12] HUANG, H. H., LI, S., SZALAY, A., AND TERZIS, A. Performance modeling and analysis of flash-based storage devices. In *Mass Storage Systems and Technologies (MSST), IEEE 27th Symposium on* (2011), IEEE, pp. 1–11.
- [13] JOHNSON, D. S., DEMERS, A., ULLMAN, J. D., GAREY, M. R., AND GRAHAM, R. L. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing* 3, 4 (1974), 299–325.
- [14] KIM, E. Ssd performance-a primer: An introduction to solid state drive performance, evaluation and test. Tech. rep., Storage Networking Industry Association, 2013.
- [15] KIM, J., LEE, D., AND NOH, S. H. Towards slo complying ssds through ops isolation. In *Proceedings of the 13th USENIX Conference on File and Storage Technologies* (2015), USENIX Association, pp. 183–189.
- [16] LU, H., SALTAFORMAGGIO, B., KOMPELLA, R., AND XU, D. vfair: latency-aware fair storage scheduling via per-io cost-based differentiation. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (2015), ACM, pp. 125–138.

- [17] MASHTIZADEH, A., CELEBI, E., GARFINKEL, T., CAI, M., ET AL. The design and evolution of live storage migration in vmware esx. In *USENIX ATC* (2011), vol. 11, pp. 1–14.
- [18] OPENSTACK, L. Openstack cinder filter scheduler. http://docs.openstack.org/developer/cinder/devref/filter_scheduler.html, 2013.
- [19] OUYANG, J., LIN, S., JIANG, S., HOU, Z., WANG, Y., AND WANG, Y. Sdf: Software-defined flash for web-scale internet storage systems. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems* (2014), ACM, pp. 471–484.
- [20] PARK, N., AHMAD, I., AND LILJA, D. J. Romano: autonomous storage management using performance prediction in multi-tenant datacenters. In *Proceedings of the Third ACM Symposium on Cloud Computing* (2012), ACM, p. 21.
- [21] POSITION, S. T. Solid state storage (sss) performance test specification (pts) enterprise. Tech. rep., Storage Networking Industry Association, 2013.
- [22] VOELLM, T. Useful io profiles for simulating various workloads. <http://blogs.msdn.com/b/tvoellm/archive/2009/05/07/useful-io-profiles-for-simulating-various-workloads.aspx>, 2009.
- [23] WANG, A., VENKATARAMAN, S., ALSPAUGH, S., KATZ, R., AND STOICA, I. Cake: enabling high-level slos on shared storage systems. In *Proceedings of the Third ACM Symposium on Cloud Computing* (2012), ACM, p. 14.
- [24] WIKI, T. K. Tkperf benchmark toolkit. <https://www.thomas-krenn.com/en/wiki/Tkperf>.
- [25] WOOD, T., SHENOY, P. J., VENKATARAMANI, A., AND YOUSIF, M. S. Black-box and gray-box strategies for virtual machine migration. In *NSDI* (2007), vol. 7, pp. 17–17.
- [26] YAO, Z., PAPAPANAGIOTOU, I., AND CALLAWAY, R. D. Slaware resource scheduling for cloud storage. In *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on* (2014), IEEE, pp. 14–19.
- [27] YAO, Z., PAPAPANAGIOTOU, I., AND CALLAWAY, R. D. Multi-dimensional scheduling in cloud storage systems. In *Communications, 2015 IEEE 3rd International Conference on* (2015), IEEE.
- [28] ZHU, T., TUMANOV, A., KOZUCH, M. A., HARCHOL-BALTER, M., AND GANGER, G. R. Prioritymeister: Tail latency qos for shared networked storage. In *Proceedings of the ACM Symposium on Cloud Computing* (2014), ACM, pp. 1–14.